

Bounded Computational Capacity Equilibrium*

Penélope Hernández[†] and Eilon Solan[‡]

August 17, 2010

Abstract

We study repeated games played by players with bounded computational power, where, in contrast to Abreu and Rubinstein [1], the memory is costly. We prove a folk theorem: the limit set of equilibrium payoffs in mixed strategies, as the cost of memory goes to 0, includes the set of feasible and individually rational payoffs. This result stands in sharp contrast to [1], who proved that when memory is free, the set of equilibrium payoffs in repeated games played by players with bounded computational power is a strict subset of the set of feasible and individually rational payoffs. Our result emphasizes the role of memory cost and of mixing when players have bounded computational power.

Keyword: Bounded rationality, automata, complexity, infinitely repeated games, equilibrium.

1 Introduction

In a seminal work, Simon [21], [22] recognized the impact of bounded rationality in economic modelization of individual agents and of organizations. In the last few

*This work was conducted while the second author was visiting Universidad de Valencia. The first author thanks both the Spanish Ministry of Science and Technology and the European Feder Funds for financial support under project SEJ2007-66581 and Generalitat Valenciana (PROMETEO/2009/068). The second author thanks the Departamento de Análisis Económico at Universidad de Valencia for the hospitality during his visit. The authors thank Elchanan Ben Porath, Ehud Kalai and Ehud Lehrer for their suggestions. The work of Solan was partially supported by ISF grant 212/09.

[†]ERI-CES and Departamento de Análisis Económico, Universidad de Valencia. Campus de Los Naranjos s/n, 46022 Valencia, Spain.

[‡]Department of Statistics and Operations Research, School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel. eilons@post.tau.ac.il

decades an expanding literature studied the implementation cost of strategies in strategic interactions (see, e.g., Rubinstein [19], Chatterjee and Sabourian [5]). One particular research question deals with achieving a target outcome as a collusion, co-operation, or bargaining outcome by non-sophisticated agents (see, e.g., Chatterjee and Sabourian [4], Sabourian [20], Gale and Sabourian [6], and Maenner [9]).

One common way to model players with bounded rationality is by restricting them to strategies that can be implemented by finite state machines, or automata. The game theoretic literature on repeated games played by finite automata can be roughly divided into two categories. On the one hand, an extensive literature (e.g., Kalai [8], Ben Porath [3], Piccione [16], Piccione and Rubinstein [17], Neyman [10], [11], [12], Neyman and Okada [13], [14], [15], Zemel [23]) study games where the memory size of the two players is determined exogenously, so that each player can deviate only to strategies with the given memory size. On the other hand, Rubinstein [18], Abreu and Rubinstein [1] and Banks and Sundaram [2] study games where the players have lexicographic preferences: each player tries to maximize her payoff, and subject to that she tries to minimize her memory size. Thus, it is assumed that memory is free, and a player would deviate to significantly more complex strategy if that would increase her profit by one cent.

In practice, the level of complexity that players can use in their strategies is not known in advance, either because players do not know each other computational power, because players may increase their computational power if they realize that such an increase is beneficial, or because players may decrease their computational power if the loss caused by this decrease is compensated by the reduced expenses due to this decision.

In the present paper we take a more pragmatic point of view than the two approaches mentioned above, and we study repeated games played by boundedly rational players, when the computational power is costly.

As a motivating example, consider employees' training for a new job. The training period enables the employee to cope with situations that he may encounter in the future. The longer the training period, the better prepared will be the employee, thereby increasing the employer's profit.

Once the employee starts working, he follows the instructions that he learned, and so we can model the employee as a finite state machine. The training period dictates the size of the machine, that is, the number of its states, and the training itself determines how the machine behaves in various situations. Because training is costly, the employer will try to balance between the length of the training period and the gains from extended training.

When employees of different employers interact, say a salesperson and a buyer, the evolution of the interaction is dictated by their training. A salesperson, say, may interact with buyers of different firms, who undertook different training programs,

and therefore follow different finite state machines. Therefore he has some uncertainty regarding the finite state machine that the buyer will follow, so that in fact he faces a mixed strategy.

The employers, who plan the training of their respective employees, then face a game, where each tries to teach her employees the techniques that best cope with the techniques taught by the other employer. As salespersons and buyers interact repeatedly, the situation can be modelled as a repeated game played by finite state machines, where the goal of each player is to maximize some combination of the long-run average payoff and the cost of training.

To capture situations like the one in the example, we assume for simplicity that the players have additive utility: the utility of a player is the sum of her long-run average payoff and the cost of her computational power. Formally, for every positive real number c , we say that the vector $x \in \mathbb{R}^2$ is a *c-Bounded Computational Capacity equilibrium* (hereafter, BCC for short) if it is an equilibrium when the utility of each player is the difference between her long-run average payoff and c times the size of its finite state machine.

A payoff vector $x \in \mathbb{R}^2$ is a *BCC equilibrium payoff* if it is the limit, as c goes to 0, of payoffs that correspond to c -bounded computational capacity payoffs, and the cost of the machines used along the sequence converges to 0.

Interestingly, the definition does not imply that the set of BCC equilibrium payoffs is a subset, nor a super set, of the set of Nash equilibrium payoffs.

Our main result is a folk theorem: in two player games, every feasible and individually rational (w.r.t. the min-max value in pure strategies) payoff vector is a BCC equilibrium payoff.

Our proof is constructive: we explicitly construct equilibrium strategies. The equilibrium play is composed of three phases. The first phase, that on the equilibrium path is played only once, is a punishment phase; in this phase each player plays a strategy that punishes the other player, that is, an action that attains the min-max value in pure strategies of the opponent. As in [1], it is crucial to have the punishment phase on the equilibrium path; otherwise, players can use smaller machines, that cannot implement punishment and lower the cost of their machines. However, if a machine cannot implement punishment, there is nothing that will deter the other player from deviating. The second phase, called the babbling phase, is also played only once on the equilibrium path. In this phase the players play a predetermined sequence of action pairs. In the third phase, called the regular phase, the players play repeatedly a predetermined periodic sequence of action pairs that approximates the desired target payoff. To implement this phase, the players re-use states that were used in the babbling phase. In fact, the role of the babbling phase is to enable one to embed the regular phase within it, and its structure is designed to simplify complexity calculations. It is long enough to ensure that with only low probability a player can

correctly guess which of the states in the other player's machine are re-used.

One can describe the equilibrium path by imagining the following meeting between two strangers. At first, the strangers exchange threats and vivid descriptions of what each one will do to the other if the other does not behave as desired. After they prove to each other that they can execute punishment, they indulge in a long small-talk. Finally, they go to business, and implement the desired outcome.

Our paper is closely related to Abreu and Rubinstein [1], where a characterization of the set of equilibrium payoffs is provided when the players have a lexicographic preference: subject to maximizing her long-run average payoff each player wishes to minimize the complexity of the finite state machine that implements her strategy. The main result of [1] is that the set of equilibrium payoffs is the set of all feasible and individually rational payoffs (relative to the min-max value in pure strategies) that can be supported by coordinated play. The main message of [1] is that a folk theorem does not obtain: the set of equilibrium payoffs may be strictly smaller than the set of feasible and individually rational payoffs. Our result shows that two properties of the model of [1] drive their result. First, [1] assumes that computational power is costless, so that players will deviate to a prohibitively large automaton to gain a cent. This is in contrast to our model, where computational power is costly. Second, [1] restricts the players to pure strategies, whereas we allow the players to use mixed strategies.

Abreu and Rubinstein [1] point at a difficulty in using mixed strategies in games played by players with bounded computational power: mixing is a complex operation, and players with bounded computational power will prefer to use a pure strategy than a mixed strategy, thereby saving the cost of mixing. We argue that there are at least two interpretations of the model where the use of mixed strategies is natural. First, it may happen that the agent playing the game is limited, whereas the player who chooses the strategy for the agent does not have limits on her computational power. Thus, the complexity of computing the strategy played by the agent can be large, and include mixing, while the complexity of implementing this strategy should be low. Second, a player may not know the identity of the agent whom her own agent is going to face, and therefore she does not know the pure simple strategy which that agent is going to use. Alternatively, the other agents whom her agent is going to face may use different pure strategies. Thus, the player may assume that the other player randomly chooses her simple strategy. In our construction the role of mixing is to hide the strategy that each agent uses. Whereas in [1] the players use pure strategies to reduce their computational power, which leads to a significantly smaller set of equilibrium payoffs, mixing allows the players to use once again complex strategies, and the folk theorem is restored.

The rest of the paper is organized as follows. Section 2 presents the model and the main result. The construction of a mixed equilibrium strategy for both players

in the particular case of the Prisoner's Dilemma is presented in Section 3. In Section 4 we explain how the construction is adapted for general two-player games.

2 The Model and the Main Result

In this section we define the model, including the concepts of automata, repeated games, and strategies implementable by an automaton; we describe our solution concept of Bounded Computational Capacity equilibrium, and we state the main result.

2.1 Repeated Games

A two-player *repeated game* is given by (1) two finite action sets A_1 and A_2 for the two players, and (2) two payoff functions $u_1 : A_1 \times A_2 \rightarrow \mathbb{R}$ and $u_2 : A_1 \times A_2 \rightarrow \mathbb{R}$ for the two players.

The game is played as follows. At every stage t , each player $i \in \{1, 2\}$ chooses an action $a_i^t \in A_i$, and receives the stage payoff $u_i(a_1^t, a_2^t)$. The goal of each player is to maximize its long-run average payoff $\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{j=1}^t u_i(a_1^j, a_2^j)$, where $\{(a_1^j, a_2^j), j \in \mathbb{N}\}$ is the sequence of action pairs that were chosen by the players.¹ A *pure strategy* of player i is a function that assigns an action in A_i to every finite history $h \in \cup_{t=0}^{\infty} (A_1 \times A_2)^t$. A *mixed strategy* of player i is a probability distribution over pure strategies.

2.2 Automata

A common way to model a decision maker with bounded computational capacity is as an automaton, which is a finite state machine whose output depends on the current state, and whose evolution depends on the current state and on its input (see, e.g., Neyman [10] and Rubinstein [18]). Formally, an *automaton* P is given by (1) a finite state space Q , (2) a finite set I of inputs, (3) a finite set O of outputs, (4) an output function $f : Q \rightarrow O$, (5) a transition function $g : Q \times I \rightarrow Q$, and (6) an initial state $q^* \in Q$.

Denote by q^t the automaton's state at stage t . The automaton starts in state $q^1 = q^*$, and at every stage t , as a function of the current state q^t and the current input i^t , the output of the automaton $o^t = f(q^t)$ is determined, and the automaton moves to a new state $q^{t+1} = g(q^t, i^t)$.

The *size* of an automaton P , denoted by $|P|$, is the number of states in Q . Below we will use strategies that can be implemented by automata; in this case the size of the automaton measures the complexity of the strategy.

¹In general this limit need not exist. Our solution concept will take care of this issue.

2.3 Strategies Implemented by Automata

Fix a player $i \in \{1, 2\}$. An automaton P whose set of inputs is the set of actions of player $3-i$ and set of outputs is the set of actions of player i , that is, $I = A_{3-i}$ and $O = A_i$, can implement a pure strategy of player i . Indeed, at every stage t , the strategy plays the action $f(q^t)$, and the new state of the automaton $q^{t+1} = g(q^t, a_{3-i}^t)$ depends on its current state q^t and on the action a_{3-i}^t that the other player played at stage t . For $i = 1, 2$, we denote an automaton that implements a strategy of player i by P_i . We denote by \mathcal{P}_i^m the set of all automata with m states that implement pure strategies of player i .

When the players use arbitrary strategies, the long-run average payoff needs not exist. However, when both players use strategies that can be implemented by automata, say P_1 and P_2 of sizes p_1 and p_2 respectively, the evolution of the automata follows a Markov chain with $p_1 \times p_2$ states, and therefore the long-run average payoff exists. We denote this average payoff by $\gamma(P_1, P_2) \in \mathbb{R}^2$.

A mixed automaton M is a probability distribution over pure automata². A mixed automaton corresponds to the situation in which the automaton that is used is not known, and there is a belief over which automaton is used. A mixed automaton defines a mixed strategy: at the outset of the game, a pure automaton is chosen according to the probability distribution given by the mixed automaton, and the strategy that the pure automaton defines is executed.

We will use only mixed automata whose support is pure automata of a given size m . Denote by \mathcal{M}_i^m the set of all mixed automata whose support is automata in \mathcal{P}_i^m , and by $\mathcal{M}_i = \cup_{m \in \mathbb{N}} \mathcal{M}_i^m$ the set of all mixed automata whose support contains automata of the same size. If $M_i \in \mathcal{M}_i^m$, we say that m is the size of the automaton M_i . Thus, the size of a mixed automaton refers to the *size* of the pure automata in its support (and not, for example, to the *number* of pure automata in its support). If we interpret each pure automaton as an agent's type, and a mixed automaton as the type's distribution in the population, then the size of the mixed automaton measures the complexity of an individual agent, and not the type diversity in the population.

When both players use mixed strategies that can be implemented by mixed automata, the expected long-run average payoff exists; it is the expectation of the long-run average payoff of the (pure) automata that the players play:

$$\gamma(M_1, M_2) := \mathbf{E}_{M_1, M_2}[\gamma(P_1, P_2)].$$

²To emphasize the distinction between automata and mixed automata, we call the former *pure automata*.

2.4 Bounded Computational Capacity Equilibrium

In the present section we study games where the utility function of each player takes into account the complexity of the strategy that she uses.

Definition 1 Let $c > 0$. A pair of mixed automata (M_1, M_2) is a c -BCC equilibrium, if it is a Nash equilibrium for the utility functions $U_i^c(M_1, M_2) = \gamma_i(M_1, M_2) - c|M_i|$, $i \in \{1, 2\}$.

If the game has an equilibrium in pure strategies, then the pair of pure automata (P_1, P_2) , both with size 1, that repeatedly play the equilibrium actions of the two players, is a c -BCC equilibrium, for every $c > 0$.

The min-max value of player i in pure strategies in the one-shot game is

$$v_i := \min_{a_{3-i} \in A_{3-i}} \max_{a_i \in A_i} u_i(a_i, a_{3-i}).$$

An action a_{3-i} that attains the minimum is termed a *punishing* action of player $3-i$.

To get rid of the dependency of the constant c we define the concept of a *BCC equilibrium payoff*. A payoff vector x is a *BCC equilibrium payoff* if it is the limit, as c goes to 0, of the payoff that corresponds to c -BCC equilibria.

Definition 2 A payoff vector $x = (x_1, x_2)$ is a BCC equilibrium payoff if for every $c > 0$ there is a c -BCC equilibrium $(M_1(c), M_2(c))$ such that $\lim_{c \rightarrow 0} U^c(M_1(c), M_2(c)) = x$ and $\lim_{c \rightarrow 0} cM_i(c) = 0$.

The condition $\lim_{c \rightarrow 0} U^c(M_1(c), M_2(c)) = x$ in the definition of a BCC-equilibrium payoff ensures that x can be supported as an equilibrium, while the condition $\lim_{c \rightarrow 0} cM_i(c) = 0$ ensure that the cost of the automata that support this equilibrium is negligible. In particular, the limit of the long-run average payoffs also converges to x : $\lim_{c \rightarrow 0} \gamma(M_1(c), M_2(c)) = x$.

It follows from the discussion above that every pure equilibrium payoff is a BCC equilibrium payoff. Using Abreu and Rubinstein's [1] proof, one can show that any individually rational payoff (relative to the min-max value in pure strategies) that can be generated by coordinated play is a BCC equilibrium payoff. For the formal statement, assume w.l.o.g. that $|A_1| \leq |A_2|$.

Theorem 3 (Abreu and Rubinstein, 1988) Let $\sigma : A_1 \rightarrow A_2$ be a one-to-one function. Then any payoff vector x in the convex hull of $\{u(a_1, \sigma(a_1)), a_1 \in A_1\}$ that satisfies $x_i > v_i$ for $i = 1, 2$ is a BCC equilibrium payoff.

2.5 The Main Result

The set of feasible payoff vectors is

$$F := \text{conv}\{u(a), a \in A_1 \times A_2\}.$$

The set of *strictly individually rational payoff vectors* (relative to the min-max value in pure strategies) is

$$V := \{x = (x_1, x_2) \in \mathbb{R}^2 : x_1 > v_1, x_2 > v_2\}.$$

Our main result is the following folk theorem, that states that every feasible and strictly individually rational payoff vector is a BCC equilibrium payoff.

Theorem 4 *Every vector in $F \cap V$ is a BCC equilibrium payoff.*

Observe that Theorem 4 is not a characterization of the set of BCC equilibrium payoffs, because it does not rule out the possibility that a feasible payoff that is not individually rational (relative to the min-max value in pure strategies) is a BCC equilibrium payoff. That is, we do not know whether threats of punishments by a mixed strategy in the one-shot game can be implemented in a BCC equilibrium.

Theorem 4 stands in sharp contrast to the main message of Abreu and Rubinstein [1], where it is proved that lexicographic preferences, which is equivalent to an infinitesimal cost function c , implies that in equilibrium players follow coordinated play, so that the set of equilibrium payoffs is sometimes smaller than the set of feasible and individually rational payoffs. Our study shows that the result of Abreu and Rubinstein [1] hinges on two assumptions: (a) memory is costless, and (b) the players use only pure automata. Once we assume that memory is costly, and that players may use mixed automata, the set of equilibrium payoffs dramatically changes.

2.6 Comments and Discussion

2.6.1 On the definition of BCC equilibria

The definition of BCC equilibrium is analog to the definition of Nash equilibrium; in both we ask whether a specific behavior (that is, a pair of strategies) is stable. Thus, in a c -BCC equilibrium we assume that each player already has an automaton with which she is going to play the game, and we ask whether playing this automaton is the best response given the automaton that the other player is going to use. As in the definition of Nash equilibrium, we do not ask how the players arrived at these automata, and we do not restrict the sizes of these automata (though the memory cost does bound the maximal size of automaton that the players will use). In principle it may well

be that some BCC equilibrium payoff can be supported only with prohibitively large automata, which we would like to rule out. That is, we may want to add the size of the automata that the players use to the definition itself. In our construction (see the proof of Theorem 4), to support a c -BCC equilibrium payoff that is close to some target payoff x we use two automata of similar sizes; the size of the automaton is related to both c and to the level of approximation to the target payoff: as c gets closer to 0, and as the c -BCC equilibrium payoff gets closer to x , we use larger automata.

2.6.2 BCC equilibria and Nash equilibria

Theorem 4 states that every feasible and individually rational (w.r.t. the max-min value in pure strategies) payoff vector is a BCC equilibrium payoff. This theorem does not rule out the possibility that there would be a payoff vector that is *not* individually rational that would still be a BCC equilibrium; that is, a BCC equilibrium payoff need not be a Nash equilibrium payoff. The theorem also does not rule out the possibility that some payoff vector that is individually rational w.r.t. the max-min value in *mixed* strategies, but not individually rational w.r.t. the max-min value in pure strategies, would not be a BCC equilibrium payoff, so that a Nash equilibrium payoff need not be a BCC equilibrium payoff.

Moreover, in zero-sum games it is not clear whether there is a unique BCC equilibrium payoff. If in zero-sum games there always is a unique BCC equilibrium payoff, then this quantity can be called the *BCC value* of the game. However, it is possible that in zero-sum games there will be more than one BCC equilibrium payoff, in which case even in this class of games, the outcome will crucially depend on the relative computational power the players have.

2.6.3 A more general definition of a BCC equilibrium

The definition of c -BCC equilibrium assumes that the utility of each player is additive, and that the memory cost is linear in the memory size. There are applications where the utility function U_i has a different form.

- Players may disregard the memory cost, but be bounded by the size of memory that they use.

$$U_i(M_1, M_2) = \begin{cases} \gamma_i(M_1, M_2) & |M_i| \leq k_i, \\ -\infty & |M_i| > k_i. \end{cases}$$

This situation occurs, e.g., when players are willing to invest huge amount of money even if the profit is low, but the available technology does not allow

them to increase their memory size beyond some limit. Such situation may occur, e.g., in the area of code breaking, where countries invest large sums of money to be able to increase the number of codes of other countries that they break, and they are only bounded by technological advances.

- Memory is costly, yet players do not save money by reducing their memory size. That is, a pair of mixed automata (M_1, M_2) is a c -BCC equilibrium if for each $i \in \{1, 2\}$ and for every pure automaton $P_i \in M_i$ one has $\gamma_i(M_i, M_{3-i}) \geq \gamma_i(P_i, M_{3-i})$, and, if $P_i > M_i$, one has $\gamma_i(M_i, M_{3-i}) \geq \gamma_i(P_i, M_{3-i}) - c(|P_i| - |M_i|)$. This situation occurs, e.g., when the players are organizations whose size cannot be reduced.

It may be of interest to study the set of equilibrium payoffs for various utility functions U_i , and to see whether and how this set depends on the shape of this function.

2.6.4 More than two players

The concept of BCC equilibrium payoff is valid to games with any number of players. However, Theorem 4 holds only for two-player games. One crucial point in our construction is that if a deviation is detected, a player is punished for a long (yet finite) period of time by a punishing action. When there are more than two players, the punishing action of, say, player 1 against player 2 may be different than the punishing action of player 1 against player 3. It is not clear how to construct an automaton that can punish each of the other players, if necessary, and such that all these memory cells will be used on the equilibrium path.

2.6.5 BCC equilibria in one-shot games

The concept of BCC equilibrium that we presented here applies to repeated games. However, the concept can be naturally adapted to one-shot games as well³. For example, consider the following game, that appears in Halpern and Pass [7]. Player 1 chooses an integer n and tells it to player 2; player 2 has to decide whether n is a prime number or not, winning 1 if she is correct, losing 1 if she is incorrect. Plainly the value of this game is 1: player 2 can check whether the choice of player 1 is a prime number. However, as there is no efficient algorithm to check whether an integer is a prime number, it is not clear whether in practice risk-neutral people would be willing to participate in this game as player 2.

³We thank Ehud Kalai for drawing our attention to this issue.

The concept of BCC equilibrium can be applied in such situations, and one can study the set of BCC equilibrium payoffs, and how this set depends on the relative memory cost of the two players.

In the context of the Computer Science literature one could conceive of an analog solution concept, where automata are replaced by Turing machines, and the memory size is replaced by the length of the machine's tape.

3 BCC Equilibria in the Prisoner's Dilemma

In the present section we prove Theorem 4 for the Prisoner's Dilemma. The construction in this case contains all the ingredients of the general case, yet the simplicity of the Prisoner's Dilemma allows one to concentrate on the main aspects of the construction. In Section 4 we indicate how to generalize this basic construction to general two-player repeated games.

The Prisoner's Dilemma is the two-player game depicted in Figure 1, where each player has two actions: $A_1 = A_2 = \{Cooperate, Defect\}$.

		Player 2	
		D	C
Player 1	D	1, 1	4, 0
	C	0, 4	3, 3

Figure 1: The Prisoner's Dilemma.

The min-max level of each player is 1, and the punishing action of each player is D . The set of feasible and (weakly) individually rational payoffs appear in Figure 2. It is equal to the quadrilateral W with extreme points $(1, 1)$, $(1, 3\frac{2}{3})$, $(3, 3)$ and $(3\frac{2}{3}, 1)$.

Figure 2: The feasible and individually rational payoffs in the Prisoner's Dilemma.

We now show that every feasible and individually rational payoff vector x is a BCC equilibrium payoff. In the construction we do not use the special structure of the payoff matrix; all we use is that each player has two actions, and that D is the punishing action of both players.

Observe that each point in W can be written as a convex combination of three vectors in the payoff matrix, $(3, 3)$, $(1, 1)$, and either $(0, 4)$ or $(4, 0)$. Assume w.l.o.g. that the latter holds, so that

$$x = \alpha_1(1, 1) + \alpha_2(4, 0) + \alpha_3(3, 3), \quad (1)$$

where $\alpha_1 + \alpha_2 + \alpha_3 = 1$ and $\alpha_1, \alpha_2, \alpha_3 \geq 0$.

Our goal is to define two sequences of mixed automata $(M_1(k))_k$ and $(M_2(k))_k$, that support x as a BCC equilibrium payoff: the long-run average payoff under $(M_1(k), M_2(k))$ will converge to x . The road-map of the proof is as follows. We fix $k \in \mathbb{N}$, and we define a play path ω^* that depends on k and that will be the equilibrium path under $(M_1(k), M_2(k))$ (Section 3.1). We then calculate a lower bound to the complexity of the play path for each player (the complexity is of the order k^3 , see Section 3.2). Recall that the complexity of a play path w.r.t. a player is the size of the smallest automaton for that player that can implement this play path, provided the other player follows her part in the play path. We then construct, for each player, a family of pure automata with this smallest size that implement the play path (Sections 3.3 and 3.4). We let the mixed automaton of each player choose one of these pure automata, and finally we prove that each of these mixed automata is a $z(k)$ -BCC best reply against the other, where $\lim_{k \rightarrow \infty} z(k) = 0$ (see Sections 3.5 and 3.6).

3.1 The Equilibrium Play

We fix throughout a natural number k , sufficiently large to satisfy several conditions that will be set in the sequel. Let k_0 be the largest integer that satisfies $(k_0)^2 + k_0 \leq k$. We here define a specific play path ω^* that will be the equilibrium path.

We approximate $(\alpha_1, \alpha_2, \alpha_3)$ by rational numbers with denominator k_0 ; that is, let (k_1, k_2, k_3) be three natural numbers that satisfy (a) $k_1 + k_2 + k_3 = k_0$, and (b) $\|\frac{k_j}{k_0} - \alpha_j\| \leq \frac{1}{\sqrt{k_0}}$ for $j = 1, 2, 3$. Let k be a sufficiently large integer such that there are at least k_2 prime numbers larger than k_2 and smaller than $k - k_1$. Because the number of prime numbers smaller than k is approximately $\frac{k}{\ln(k)}$, k is of the order⁴ of $k_2 \ln(k_2)$.

Let ω_0 be the following play of length k_0 that generates a payoff close to x :

$$\omega_0 = k_1 \times (D, D) + k_2 \times (D, C) + k_3 \times (C, C) \quad (2)$$

$$= \underbrace{(D, D), \dots, (D, D)}_{k_1 \text{ times}}, \underbrace{(D, C), \dots, (D, C)}_{k_2 \text{ times}}, \underbrace{(C, C), \dots, (C, C)}_{k_3 \text{ times}}. \quad (3)$$

Here, the notation $n \times a$ means a repetition of n times the action pair a , and $\omega_1 + \omega_2$ means the concatenation of ω_1 and ω_2 . Because of the choice of (k_1, k_2, k_3) , the average payoff along ω_0 is $\frac{12}{\sqrt{k_0}}$ -close to x .

Let ω^* be the play path that consists of the followings three parts:

⁴In (b) we require that $\|\frac{k_j}{k_0} - \alpha_j\| \leq \frac{1}{\sqrt{k_0}}$ rather than $\|\frac{k_j}{k_0} - \alpha_j\| \leq \frac{1}{k_0}$, to accommodate the case $\alpha_3 = 0$. If $\alpha_3 = 0$, with the latter requirement we would have $k_3 \in \{0, 1\}$, and there would not be k_2 prime numbers between k_2 and $k - k_1$.

- A *punishment phase* that consists of k^3 times playing (D, D) .
- A *babbling phase*, that consists of $2k + 1$ blocks: in odd blocks (except the last one) the players play k times (C, C) , in even block they play k times (D, D) , and in the last block the players play $k + 1$ times (C, C) .
- A *regular play*, in which the players repeatedly play ω_0 .

Formally, the play path ω^* is:

$$\omega^* = \underbrace{k^3 \times (D, D)}_{\text{Punishment}} + \underbrace{\sum_{n=1}^k (k \times (C, C) + k \times (D, D)) + (k + 1) \times (C, C)}_{\text{Babbling}} + \underbrace{\sum_{n=1}^{\infty} \omega_0}_{\text{Regular}} .$$

The roles of the three phases are as follows.

- As in Abreu and Rubinstein [1], the punishment phase ensures that punishment is on the equilibrium path. Because the players minimize their automaton size, subject to maximizing their payoff, if the punishment phase was off the equilibrium path, players could save states by not implementing it. But if a player does not implement punishment, the other player may safely deviate, knowing that she will not be punished. In our construction, detectable deviations of the other player will lead the automaton to restart and re-implement ω^* , thereby initiating a long punishment phase. The length of the punishment phase, k^3 , is much longer than the babbling phase to ensure that the punishment is severe.
- The importance of the babbling phase is that it allows us to build up the mixed strategy equilibrium. To reach any equilibrium payoff in the convex hull, players need to implement sequences of action pairs. Some of them could be played by means of some previously used states. Nevertheless this construction may fail due to the possible deviation (without punishment) of the opponent. In order to avoid this weakness, it must be concealed the position of such re-used states. It is here where the use of the mixed strategy plays a decisive role: to hide the chosen pure strategy. This set of pure strategies will be characterised by the location of the re-used states within a convenient set of states. In our construction this is implemented by the babbling phase.

The babbling phase which serves two purposes. First, because it is coordinated, it is not difficult to calculate its complexity. Second, it is sufficiently long, so that to implement the regular phase one does not need new states, but rather one can re-use states that implement the babbling phase. Moreover, its long lengths ensures that, if the states that are re-used are chosen randomly, to find which

states are re-used with non-negligible probability the other player must use a very large automaton: to profit by deviating the other player needs to search for the re-used states, a task that requires a significantly larger automaton than the one she currently uses.

- On the equilibrium path the regular play will be played repeatedly, so that the long-run average payoff will be the average payoff along ω_0 , which is close to x .

3.2 The complexity of ω^*

Let ω be a (finite or infinite) sequence of action pairs. We say that a mixed automaton M_i of player i is *compatible* with the play ω if, when the other player $3-i$ plays her part in ω , the automaton generates the play of player i in ω (with probability 1). Plainly, different automata may be compatible with the same sequence ω . The *complexity* of ω w.r.t. player i is the size of the smallest automaton of player i that is compatible with ω . This concept was first defined and studied by Neyman [12], who also provided a simple way to calculate it.

Our goal now is to calculate the complexity of ω^* w.r.t. the two players.

Lemma 5 *The complexity of ω^* w.r.t. player 1 is $k^3 + 2k^2 + 1$, and its complexity w.r.t. player 2 is $k^3 + 2k^2 + k + 1$.*

In the rest of this subsection we prove that the complexity of ω^* w.r.t. each of the players is at least the quantities given in Lemma 5. In the next two subsections we provide an automaton for player 1 (resp. for player 2) with size $k^3 + 2k^2 + 1$ (resp. $k^3 + 2k^2 + k + 1$) that is compatible with ω^* , thereby completing the proof of Lemma 5.

We start by recalling Neyman's [12] characterization for the complexity of a play w.r.t. a player.

Denote by ω_t the sequence ω after deleting the first $t - 1$ elements from the sequence.⁵ Given a sequence of action pairs ω , finite or infinite, define an equivalence relation on the set of natural numbers \mathbb{N} as follows: t is equivalent (for player i) to t' if any automaton of player i that is compatible with ω_t is also compatible⁶ with $\omega_{t'}$. Denote this equivalence relation by $\sim_{\omega,i}$. Neyman [12] proved that the complexity of ω w.r.t. a player is the number of equivalence classes in this equivalence relation.

⁵If ω is a finite play, and t is larger than the length of ω , then ω_t is an empty sequence of action pairs.

⁶In particular, the empty play is equivalent to any other play.

3.2.1 The complexity of ω^* w.r.t. player 1 is at least $k^3 + 2k^2 + 1$

The complexity of a sequence is at least the complexity of any of its subsequences (Lemma 2 in Neyman [12]). To bound the complexity of ω^* w.r.t. player 1 we calculate the complexity w.r.t. player 1 of the following prefix $\omega^*(1)$ of ω^* :

$$\omega^*(1) = k^3 \times (D, D) + \sum_{n=1}^k (k \times (C, C) + k \times (D, D)) + (k+1) \times (C, C).$$

In $\omega^*(1)$ the players play a coordinated play, i.e., there exists a one-to-one relationship between the actions played by player 1 and the actions played by player 2: in every stage either both players play C or both players play D . Therefore, for every t , any automaton of player 1 that is compatible with $\omega_t^*(1)$ can ignore the actions of player 2. Consequently, an automaton of player 1 generates a deterministic sequence of actions. This implies that if $t_1 < t_2$, and t_1 and t_2 are equivalent (w.r.t. $\sim_{\omega^*(1),1}$), then $\omega_{t_2}^*(1)$ is a prefix of $\omega_{t_1}^*(1)$.

Because a sequence of $k+1$ times C appears only at the end of the sequence $\omega^*(1)$, it follows that $\omega_{t_2}^*(1)$ is not a prefix of $\omega_{t_1}^*(1)$ whenever $t_1 < t_2 \leq k^3 + 2k^2 + 1$. In particular, the complexity of ω^* to player 1 is at least $k^3 + 2k^2 + 1$.

3.2.2 The complexity of ω^* w.r.t. player 2 is at least $k^3 + 2k^2 + k + 1$

To bound the complexity of ω^* w.r.t. player 2, we calculate the complexity for player 2 of the following prefix $\omega^*(2)$ of ω^* :

$$\omega^*(2) = k^3 \times (D, D) + \sum_{n=1}^k (k \times (C, C) + k \times (D, D)) + (k+1) \times (C, C) + k_1 \times (D, D) + 1 \times (D, C).$$

Apart of the last action pair, the play path $\omega^*(2)$ consists of a coordinated play. Hence, analogously to the analysis for player 1, for every t , any automaton of player 2 that is compatible with $\omega_t^*(2)$ can ignore the actions of player 1. We now count the number of equivalence classes of the relation $\sim_{\omega^*(2),2}$. The sequence $1 \times (C, C) + k_1 \times (D, D) + 1 \times (C, C)$ appears along $\omega^*(2)$ only after $k^3 + 2k^2 + k + 1$ stages in $\omega^*(2)$. It follows that the number of equivalence classes of $\sim_{\omega^*(2),2}$ is at least $k^3 + 2k^2 + k + 1$. In particular, the complexity of ω^* to player 2 is at least $k^3 + 2k^2 + k + 1$.

3.3 An automaton M_1 for player 1

In this section we define a family of pure automata for player 1, all have size $k^3 + 2k^2 + 1$. Each automaton in the family is compatible with ω^* . This will prove that the complexity of ω^* w.r.t. player 1 is $k^3 + 2k^2 + 1$, as stated in Lemma 1. In section

3.3.5 we define a mixed automaton for player 1 that is supported by pure automata in this family and that will be part of the d -BCC equilibrium for a proper $d > 0$.

The automata in the family are parameterized by two parameters: an integer $j \in \{1, 2, \dots, k-1\}$ and a set $H = \{h_1, h_2, \dots, h_{k_2}\}$ of k_2 integers. The range of h_1, h_2, \dots, h_{k_2} will be defined in step 3 below where they are used.

Given a pair (j, H) we proceed to construct a pure automaton $P_1^{j,H}$ for player 1. For clarity of the exposition, the construction is divided into three steps. We start in step 1 by defining transitions that implement the prefix of length $k^3 + 2k^2 + 1$ of ω^* . In step 2 we add transitions that implement the next $k + k_1$ action pairs in ω^* , and in step 3 we add transitions that implement the rest of ω^* . In step 1 we will use all the states of $P_1^{j,H}$. In step 2 and 3 we will re-use states for implementing the rest of ω^* . The mixed automaton that we will define later will choose j and H randomly, to conceal the states that are re-used.

The size of the automaton $P_1^{j,H}$ that we construct is $k^3 + 2k^2 + 1$. Denote its states by the integers $Q = \{1, 2, \dots, k^3 + 2k^2 + 1\}$, where 1 is the initial state.

3.3.1 Step 1: Implementing the prefix of ω^* of length $k^3 + 2k^2 + 1$.

The prefix of length $k^3 + 2k^2 + 1$ of ω^* is:

$$\omega_1 = k^3 \times (D, D) + \sum_{n=1}^k (k \times (C, C) + k \times (D, D)) + (C, C).$$

This play consists of the punishment phase followed by k pairs of blocks, each block is made of a C -block and a D -block (both of length k). The length of ω_1 is equal to the size of the automaton, and therefore a naive implementation is to have one state for each action of player 1 in ω_1 : state $q \in Q$ will implement the q 'th action pair in ω_1 . Formally, we divide Q to three sets:

1. $Q^P = \{1, 2, \dots, k^3\}$: this is the set of all states that implement the punishment phase.
2. $Q^C = \bigcup_{n=0}^{k-1} \{k^3 + 2nk + 1, \dots, k^3 + 2nk + k\} \cup \{k^3 + 2k^2 + 1\}$: this is the set of states in all C -blocks.
3. $Q^D = \bigcup_{n=0}^{k-1} \{k^3 + 2nk + k + 1, \dots, k^3 + 2nk + 2k\}$: this is the set of states in all D -blocks.

The output function is:

$$f(q) = \begin{cases} D & q \in Q^P \cup Q^D, \\ C & q \in Q^C, \end{cases}$$

and the transition function is

$$g(q, f(q)) = q + 1, \quad 1 \leq q < k^3 + 2k^2 + 1.$$

Because the play in ω_1 is coordinated, the transition is defined only if player 2 complies with the desired play ω_1 . Figure 3 illustrates the first step in the construction of the automaton $P_1^{j,H}$. In this figure, the initial state is the dotted circle to the left, the white squares correspond to states where the action is D , and the black circles correspond to states where the action is C .

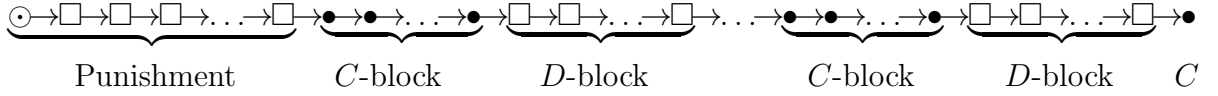


Figure 3: An implementation of ω_1 .

3.3.2 Step 2: Implementing the next $k + k_1$ action pairs.

We now add to the automaton $P_1^{j,H}$ transitions that implement the next $k + k_1$ action pairs in ω^* , which are

$$\omega_2 = k \times (C, C) + k_1 \times (D, D)$$

Here we use the parameter j . Because (a) the play ω_2 starts with $k \times (C, C)$, and (b) each C -block has length k and is followed by a D -block whose length is more than k_1 , we can use the j 'th C -block and the following D -block to implement ω_2 . Therefore, to implement ω_2 it is sufficient to add one transition to $P_1^{j,H}$, from the last state to the beginning of the j 'th C -block:

$$g(k^3 + 2k^2 + 1, C) = k^3 + 2(j - 1)k + 1.$$

Figure 4 illustrate the automaton $P_1^{j,H}$ with this additional transition.

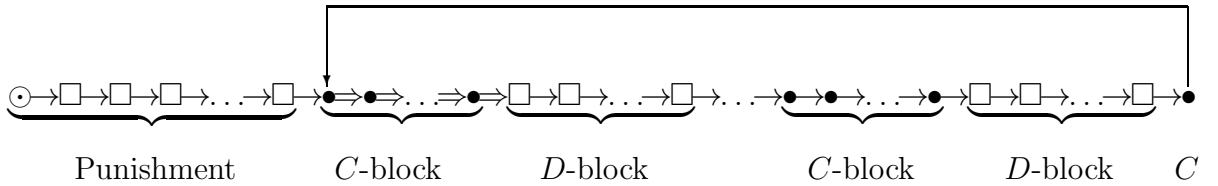


Figure 4: The automaton $P_1^{j,H}$ after the second step.

3.3.3 Step 3: Implementing the rest of ω^* .

We now add to the automaton $P_1^{j,H}$ transitions that implement the next $k_2 + k_3$ action pairs in ω^* , which are

$$\omega_3 = k_2 \times (D, C) + k_3 \times (C, C),$$

and continue to implement the regular play, which is a periodic repetition of ω_0 .

Here we use the parameter set H . To implement the k_2 repetitions of (D, C) we re-use states in a D -block, whose identity is determined by the set H . Thus, whenever in a re-used state, if player 2 plays D , the automaton $P_1^{j,H}$ assumes that the play is in the babbling phase, whereas if player 2 plays C , the automaton assumes that ω_3 is implemented. Because ω_3 comes after a sequence $k_1 \times (D, D)$, the first re-used state must be the $k_1 + 1$ state in the j 'th D -block. Because after the sequence $k_3 \times (C, C)$ the play continues with the next repetition of ω_0 , namely, with $k_1 \times (D, D)$, the sequence $k_3 \times (C, C)$ will be implemented at the end of the j 'th C -block.

Formally, assume that the set H satisfies the following two conditions:

(D1) $h_1 = k^3 + 2(j-1)k + k + k_1 + 1$, and

(D2) h_2, h_3, \dots, h_{k_2} are distinct states in Q^D , all different from h_1 .

We add the following transitions (see Figure 5):

$$g(h_n, C) = h_{n+1}, \quad 1 \leq n < k_2 - 1, \quad (4)$$

$$g(h_{k_2}, C) = k^3 + 2(j-1)k + (k - k_3). \quad (5)$$

In Figure 5, re-used states are denoted by triangles. When the automaton $P_1^{j,H}$ is at such a state it plays the action D ; if player 2 plays the action D , the transition is to the subsequent (square) state, whereas if player 2 plays C , the transition is to the next triangle state.

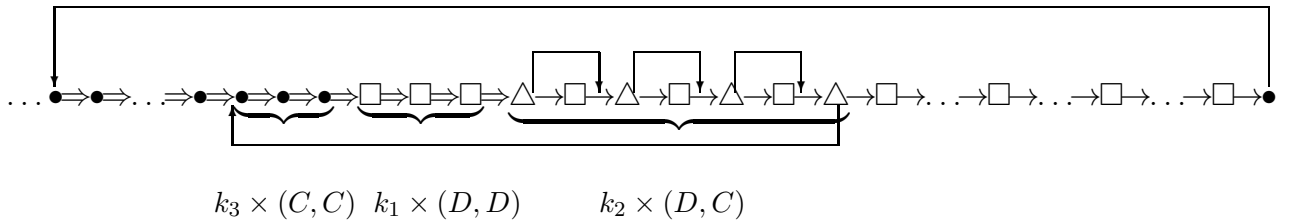


Figure 5: The j 'th C -block and D -block in $P_1^{j,H}$.

3.3.4 Last step: Deviations.

By construction, the automaton $P_1^{j,H}$ is compatible with ω^* . In particular, the complexity of ω^* w.r.t. player 1 is $k^3 + 2k^2 + 1$ as stated in Lemma 5. We now add transitions to detect deviations of player 2 as follows: all transitions that were not defined in steps 1-3 lead to state 1.

Only re-used states accept both actions of player 2; the other states accept only the action that is indicated by ω^* . Because a punishment phase of length k^3 begins in state 1, any deviation in a non re-used state is followed by a severe punishment. In the next subsection we define the mixed automaton that player 1 uses. The parameters j and H will be chosen randomly, so that to profit by deviation, player 2 will have to learn j or H , and such a learning process requires a large memory.

3.3.5 Mixed strategy

We now define the mixed automaton $M_1 = M_1(k)$ for player 1. For every n , $1 \leq n \leq k_2$, define

$$\widehat{H}_n = \{1, 1+n, 1+2n, 1+3n, \dots, 1+k_2n\},$$

and

$$H_n = \{k^3 + 2(n-1)k + k + k_1 + h : h \in \widehat{H}_n\}.$$

Thus, H_n contains k_2 states in the n 'th D -block, that are equally spaced, and the distance between each two adjacent states is n . Because $k_0 + (k_0)^2 \leq k$, there are enough states in the D -block to accommodate this construction, and the two conditions (D1) and (D2) (in page 18) are satisfied.

Let $J = \{j_1, j_2, \dots, j_{k_2}\}$ be a collection of k_2 different prime numbers in the range $\{k_2 + 1, k_2 + 2, \dots, k - k_1\}$, which exist by the choice of k . Let M_1 be the mixed automaton of player 1 that assigns a probability $\frac{1}{k_2}$ to each of the pure automata $P_1^{j_i, H_{j_i}}$.

3.4 An automaton M_2 for player 2

In this section we describe an analog construction to the one we presented in section 3.3, for a mixed automaton of player 2. We construct a family of pure automata for player 2, all of size $k^3 + 2k^2 + k + 1$, and all compatible with ω^* for player 2. As for player 1, the automata in the family depend on two parameters, an integer $j \in \{1, 2, \dots, k-1\}$ and a set H of integers.

3.4.1 Step 1: Implementing the prefix of length $k^3 + 2k^2 + k + 1$ of ω^* .

We start by implementing the prefix of length $k^3 + 2k^2 + k + 1$ of ω^* by a naive automaton with $k^3 + 2k^2 + k + 1$ states. The prefix is:

$$\omega_4 = k^3 \times (D, D) + \sum_{n=0}^{k-1} (k \times (C, C) + k \times (D, D)) + (k+1) \times (C, C),$$

and it contains the punishment phase and the babbling phase. As for player 1, we define an automaton that implements each action pair in one state. Let $Q = \{1, 2, \dots, k^3 + 2k^2 + k + 1\}$ be the set of states of the automaton, and divide Q into three sets, as follows:

1. $Q^P = \{1, 2, \dots, k^3\}$: this is the set of all states that implement the punishment phase.
2. $Q^C = \bigcup_{n=0}^{k-1} \{k^3 + 2nk + 1, \dots, k^3 + 2nk + k\} \cup \{k^3 + 2k^2 + 1, \dots, k^3 + 2k^2 + k + 1\}$: this is the set of all states in C -blocks.
3. $Q^D = \bigcup_{n=0}^{k-1} \{k^3 + 2nk + k + 1, \dots, k^3 + 2nk + 2k\}$: this is the set of all states in D -blocks.

The output function is:

$$f(q) = \begin{cases} D & q \in Q^P \cup Q^D, \\ C & q \in Q^C, \end{cases}$$

and the transition function is

$$g(q, f(q)) = q + 1, \quad 1 \leq q < k^3 + 2k^2 + k + 1.$$

3.4.2 Step 2: Implementing the next k_1 actions in ω^* .

We now add the transitions that implement the next k_1 actions in ω^* , which are $\omega_5 = k_1 \times (D, D)$. To this end, we re-use states in a D -block, and because after ω_5 player 2 plays C in ω^* , we re-use the last k_1 states that implement a D -block. So that player 1 does not know which D -block is re-used, we use the j 'th D -block. Formally,

$$g(k^3 + 2k^2 + k + 1, C) = k^3 + 2kj - k_1.$$

3.4.3 Step 3: Implementing the rest of ω^* .

We now add transitions that implement $\omega_6 = k_2 \times (D, C) + k_3 \times (C, C) + \sum_{n=1}^{\infty} \omega_0$. To implement the sequence $k_2 \times (D, C)$ we re-use states in a C -block that are determined by the set $H = \{h_1, h_2, \dots, h_{k_2}\}$. The first re-used state must be the first state in the $j + 1$ 'th C -block, and therefore $h_1 = k^3 + 2kj + 1$. Because the second part of ω_3 , that is, $k_3 \times (C, C)$, should lead to the sequence $k_1 \times (D, D)$ that starts ω_0 , the states that implement that part must be the last k_3 states in Q ; therefore we must have $h_{k_2} = k^3 + 2k^2 + k + 1 - k_3$. Finally, we require that $h_1, h_2, h_3, \dots, h_{k_2}$ are distinct states in C -blocks.

Transitions are defined as follows:

$$g(h_n, D) = h_{n+1}, \quad 1 \leq n < k_2, \quad (6)$$

$$g(h_{k_2}, D) = h_{k_2} + 1. \quad (7)$$

3.4.4 Last step: Deviations.

Finally we add transitions to handle deviations in states that are not re-used. All transitions that are not defined in steps 1-3, lead to state 1, so that such deviations initiate a long punishment phase.

3.4.5 Mixed strategy of player 2.

The definition of the mixed strategy $M_2 = M_2(k)$ is analog to that of M_1 . Recall that

$$\widehat{H}_n = \{1, 1 + n, 1 + 2n, 1 + 3n, \dots, 1 + k_2 n\},$$

and

$$H_n = \{k^3 + 2nk + k + k_1 + h : h \in \widehat{H}_n\},$$

and that J is a set of k_2 distinct prime numbers in the range $\{k_2 + 1, \dots, k - k_1\}$.

Let M_2 be the mixed automaton of player 1 that assigns a probability $\frac{1}{k_2}$ to each of the pure automata $P_2^{j_l, H_{j_l}}$.

In the following subsections we show that the sequence $(M_1(k), M_2(k))_k$ supports x as a BCC equilibrium payoff. That is, we show that (1) the expected long-run average payoff under (M_1, M_2) is $\frac{12}{\sqrt{k_0}}$ -close to x , (2) no player can profit by deviating to a smaller automaton, and (3) we bound the amount a player can profit by deviation to a larger automaton.

3.5 The expected payoff under (M_1, M_2) is close to x

By construction, the automaton M_1 (resp. M_2) is compatible with the play ω^* for player 1 (resp. player 2). Therefore, if the players use these automata the play is ω^* , and the long-run average payoff is $\frac{12}{\sqrt{k_0}}$ -close to x .

Define

$$x^* := \gamma(M_1, M_2) = \frac{k_1}{k_1 + k_2 + k_3} u(D, D) + \frac{k_2}{k_1 + k_2 + k_3} u(D, C) + \frac{k_3}{k_1 + k_2 + k_3} u(C, C).$$

3.6 (M_1, M_2) is a c -BCC Equilibrium

In this section we prove that (M_1, M_2) is a c -BCC-equilibrium, for every c that satisfies $\frac{3}{k_2 \times k^3} < c < \frac{\eta}{2k^3}$. We only prove the claims for player 2. The claims for player 1 can be proven analogously. Below we denote the state of an automaton of player i at stage t by $q_i(t)$.

For $l \in \{1, 2, \dots, k_2\}$ denote $P_1^l := P_1^{j^l, H^l}$, so that the support of M_1 is $P_1^1, P_1^2, \dots, P_1^{k_2}$. Let j^l and H^l be the parameters j and H of P_1^l , for $l = 1, 2, \dots, k_2$. Let P_2 be an arbitrary pure automaton that implements a strategy of player 2. We denote by ω^l the play that is generated under (P_1^l, P_2) .

Suppose that the players use the automata (P_1^l, P_2) . If P_2 is not compatible with ω^* for player 2, then P_1^l restarts whenever a deviation from ω^* is detected, and a punishment phase starts. Denote by t_n^l the stage at the n 'th time in which P_1^l visits state 1 when facing P_2 :

$$\begin{aligned} t_1^l &:= 1, \\ t_{n+1}^l &:= \min \{t > t_n^l : q_1(t) = 1\}, \quad n \geq 1. \end{aligned}$$

By convention, the minimum of an empty set is ∞ .

There are two scenarios where player 2 may improve her long-run average payoff. One possibility is if there exists n such that $t_n^l < \infty = t_{n+1}^l$. Then t_n^l is the last stage in which the automaton P_1^l restarts; in other words, this is the last stage in which a punishment phase starts. If the play after stage t_n^l is different than ω^* , it means that player 2 plays as if she knows j^l and/or H^l , and she might use this information to improve her payoff. Another possibility is that $(t_n^l)_{n \in \mathbb{N}}$ are finite and between two of these stages the average payoff of player 2 is higher than x_2^* (in fact, if $(t_n^l)_{n \in \mathbb{N}}$ are finite then, so that player 2 improves her payoff, the average payoff between t_n^l and $t_{n+1}^l - 1$ should be higher than x^* infinitely often).

This leads us to the following definition.

Definition 6 *The automaton P_2 fools the automaton P_1^l if either one of the following conditions hold:*

C1) There is $n_0 \in \mathbb{N}$ such that $t_{n_0}^l < \infty = t_{n_0+1}^l$ and $\omega_{n_0}^l \neq \omega^*$.

C2) $t_n^l < \infty$ for every $n \in \mathbb{N}$, and there is $n_0 \in \mathbb{N}$ such that the average payoff for player 2 between stages $t_{n_0}^l$ and $t_{n_0+1}^l - 1$ is strictly higher⁷ than x_2^* .

If condition C1 holds, we say that P_2 fools P_1^l in stages $\{t_{n_0}^l, t_{n_0}^l + 1, \dots\}$. If condition C2 holds, we say that P_2 fools P_1^l in stages $\{t_{n_0}^l, t_{n_0}^l + 1, \dots, t_{n_0+1}^l - 1\}$. In both cases we set $t_*^l = t_{n_0}^l$, and we say that at stage t_*^l player 2 starts to fool P_1^l . Denote by $R_l = \{q_2(t_*^l), q_2(t_*^l + 1), \dots, q_2(t_*^l + k^3 - 1)\}$ the k^3 states that P_2 visits at the beginning of the period in which it fools P_1^l . We will prove below that the sets $(R_l)_{l=1}^n$ are disjoint, thereby bounding from below the size of any automaton of player 2 that obtains high payoff when facing M_1 .

Neither C1 nor C2 imply that the long-run average payoff under (P_1^l, P_2) is higher than x_2^* . Yet, as the next lemma shows, the converse is true: if the long-run average payoff of player 2 under (P_1^l, P_2) exceeds x_2^* , then P_2 must have fooled P_1^l .

Lemma 7 *If P_2 does not fool P_1^l then $\gamma_2(P_1^l, P_2) \leq x_2^*$.*

Proof. Since both P_1^l and P_2 are automata, the long-run average payoff of player 2 under (P_1^l, P_2) exists. Suppose first that $t_n^l < \infty$ for every $n \in \mathbb{N}$. Because P_2 does not fool P_1^l , for every $n \in \mathbb{N}$ the average payoff of player 2 between stages t_n^l and t_{n+1}^l is at most x_2^* , and therefore $\gamma_2(P_1^l, P_2) \leq x_2^*$.

Suppose now that there is $n_0 \in \mathbb{N}$ such that $t_{n_0}^l < \infty = t_{n_0+1}^l$. Because P_2 does not fool P_1^l , we have $\omega_{n_0}^l = \omega^*$, so that the long-run average payoff of player 2 after stage $t_{n_0}^l$ is x_2^* , and the result follows. ■

Our goal is to relate the number of pure automata P_1^l that P_2 fools to the size of P_2 . In fact, we will prove that the size of P_2 is at least k^3 times the number of pure automata P_1^l that P_2 fools. To this end, we now prove that if P_2 fools both $P_1^{l_1}$ and $P_1^{l_2}$, then R_{l_1} and R_{l_2} are disjoint: the automaton of player 2 uses different states to fool each of the two automata.

Lemma 8 *Let $1 \leq l_1 < l_2 \leq k_2$. If P_2 fools both $P_1^{l_1}$ and $P_1^{l_2}$, then $R_{l_1} \cap R_{l_2} = \emptyset$.*

The subtle definition of $(j_l, H_l)_{l=1}^{k_2}$ is the key ingredient in the proof of Lemma 8. An immediate corollary of Lemma 8 is:

Corollary 9 *Denote by L_0 the number of pure automata P_1^l that P_2 fools. Then $|P_2| \geq L_0 k^3$.*

⁷Observe that in this case $t_{n_0+1}^l \geq t_{n_0}^l + k^3$. In fact, a stronger bound can be obtained.

Proof of Lemma 8.

Step 1: If P_2 fools P_1^l then the states in R_l are distinct: $|R_l| = k^3$.

At stage t_*^l the automaton P_1^l restarts; it expects the sequence $k^3 \times (D, D) + k \times (C, C)$, and none of the states $\{1, 2, \dots, k^3 + 1\}$ of P_1^l is re-used. Because this play is coordinated, its complexity is $k^3 + 1$, and therefore player 2 must use at least k^3 distinct states to implement its prefix of length k^3 .

Step 2: If R_{l_1} and R_{l_2} are not disjoint, then $q_2(t_*^{l_1} + k^3 - 1) = q_2(t_*^{l_2} + k^3 - 1)$: the last state in R_{l_1} coincides with the last state in R_{l_2} .

Suppose that R_{l_1} and R_{l_2} are not disjoint, and assume that $q_2(t_*^{l_1} + n_1) = q_2(t_*^{l_2} + n_2)$. We argue that necessarily $n_1 = n_2$. Indeed, assume to the contrary that $n_1 < n_2$. Because in the k^3 stages that follow stage $t_*^{l_1}$ the automaton $P_1^{l_1}$ plays D , and in the k^3 stages that follow stage $t_*^{l_2}$ the automaton $P_1^{l_2}$ plays D , the automaton P_2 receives the same inputs (when facing $P_1^{l_1}$ after stage $t_*^{l_1}$, and when facing $P_1^{l_2}$ after stage $t_*^{l_2}$), so that it evolves in the same way: $q_2(t_*^{l_1} + n_1 + s) = q_2(t_*^{l_2} + n_2 + s)$ for every s that satisfies $1 \leq s \leq k^3 - n_2$. Because P_2 fools $P_1^{l_1}$, the action P_2 plays in state $q_2(t_*^{l_1} + n_1 + k^3 - n_2 + 1)$ is D . Because P_2 fools $P_1^{l_2}$, the action P_2 plays in state $q_2(t_*^{l_2} + n_2 + k^3 - n_2 + 1)$ is C . But $q_2(t_*^{l_1} + n_1 + k^3 - n_2 + 1) = q_2(t_*^{l_2} + n_2 + k^3 - n_2 + 1)$, a contradiction.

Because in the first k^3 stages after visiting stage 1, both $P_1^{l_1}$ and $P_1^{l_2}$ play in the same manner (both output D), it follows that the evolution of P_2 when facing either $P_1^{l_1}$ or $P_1^{l_2}$ is the same. The claim follows.

Step 3: $R_{l_1} \cap R_{l_2} = \emptyset$.

Assume to the contrary that R_{l_1} and R_{l_2} are not disjoint. Denote by (j_1, H_1) and (j_2, H_2) the parameters (j, H) of $P_1^{l_1}$ and $P_1^{l_2}$ respectively. By Step 2, the last state in R_{l_1} coincides with the last state in R_{l_2} . Both automata $P_1^{l_1}$ and $P_1^{l_2}$ continue in the same way, until one of them observes a deviation, in which case it restarts.

Denote by $t_*^{l_1} + n$ the first stage after stage $t_*^{l_1}$ in which P_2 deviates from ω^* when facing $P_1^{l_1}$. Because $R_{l_1} = R_{l_2}$, the first stage after stage $t_*^{l_2}$ in which P_2 deviates from ω^* when facing $P_1^{l_2}$ is $t_*^{l_2} + n$. Because P_2 fools both $P_1^{l_1}$ and $P_1^{l_2}$, the state that $P_1^{l_1}$ visits in stage $t_*^{l_1} + n$ is a re-used state, as is the state that $P_1^{l_2}$ visits in stage $t_*^{l_2} + n$. Because the re-used states in $P_1^{l_1}$ are in the j_1 'th D -block, while the re-used states in $P_1^{l_2}$ are in the j_2 'th D -block, the automata $P_1^{l_1}$ and $P_1^{l_2}$ are both in re-used states only when they implement the action pairs (D, C) , that is, in the second part of the regular play ω_0 .

Let us now verify that P_2 cannot fool both $P_1^{l_1}$ and $P_1^{l_2}$. Because in a D -block both automata $P_1^{l_1}$ and $P_1^{l_2}$ play D unless a deviation is detected and a punishment phase starts, the evolution of P_2 , when facing either $P_1^{l_1}$ or $P_1^{l_2}$ is the same, as long as these automata are in the D -block. It is therefore sufficient to show that there is no sequence of actions of player 2 that differ from the play of ω^* in D -blocks, and that does not initiate a punishment phase when facing either $P_1^{l_1}$ or $P_1^{l_2}$.

Because H_1 (resp. H_2) contains k_2 numbers, equally spaced with distance j_1 (resp. j_2), the difference $h_{r_1} - h_{r_2}$ of pairs of elements in H_1 (resp. H_2) is a multiple of j_1 (resp. j_2). Because j_1 and j_2 are prime numbers larger than k_2 , the differences generated by H_1 are different than those generated by j_2 . It follows that the unique two sequences of actions of player 2 that does not initiate a punishment phase neither when facing $P_1^{l_1}$ in block j_1 nor when facing $P_1^{l_2}$ in block j_2 are (a) a repetition of k_2 times C , and (b) a repetition of $k - k_1$ times D . Because P_2 deviates from ω^* , only the sequence in (b) should be considered.

Now, in all blocks after block j_1 (resp. j_2), the automaton $P_1^{l_1}$ (resp. $P_1^{l_2}$) does not re-use states. Because P_2 fools both $P_1^{l_1}$ and $P_1^{l_2}$, it must follow the play indicated by these automata. However, because $j_1 \neq j_2$, when the first of these two automata reaches its last state, that automaton initiates a punishment phase if P_2 plays D , while the other initiates a punishment phase if P_2 plays C . This implies that if P_2 plays the sequence in (b), then it cannot fool both $P_1^{l_1}$ and $P_1^{l_2}$, as desired. ■

Recall that the min-max value in pure strategies of both players is 1. Therefore, $\min\{x_1^* - 1, x_2^* - 1\} > 0$ is the minimal difference between the target payoff x^* and the min-max value. We now prove that player 2 cannot profit by deviating to an automaton smaller than M_2 .

Lemma 10 *Let $\eta < x_2^* - 1$, and assume that k is sufficiently large so that $\frac{4}{k_2} + \frac{8}{k} < \frac{\eta}{2}$. Let P'_2 be an automaton for player 2 with size smaller than $k^3 + 2k^2 + k + 1$. Then $\gamma_2(M_1, P'_2) - c|P'_2| \leq \gamma_2(M_1, M_2) - c|M_2|$, provided $c < \frac{\eta}{2k^3}$.*

Proof. Because the complexity of ω^* w.r.t. player 2 is $k^3 + 2k^2 + k + 1$, the play under (P_1^l, P'_2) is not ω^* . By Lemma 8, and because the size of P_2 is smaller than $2k^3$, the automaton P'_2 can fool at most one of the automata (P_1^l) . Because it cannot generate ω^* , any automaton which P_2 does not fool restarts after at most $k^3 + 2k^2 + k$ stages, so that the average payoff is at most $\frac{k^3}{k^3 + 2k^2 + k} + 4\frac{2k^2 + k}{k^3 + 2k^2 + k}$. It follows that the expected payoff $\gamma_2(M_1, P'_2)$ is at most

$$4\frac{1}{k_2} + 4\frac{k_2 - 1}{k_2} \frac{2k^2 + k}{k^3 + 2k^2 + k} + \frac{k_2 - 1}{k_2} \frac{k^3}{k^3 + 2k^2 + k} \leq 1 + \frac{4}{k_2} + \frac{8}{k}.$$

Because the size of the automaton M_2 is $k^3 + 2k^2 + k + 1$, the gain of reducing the size of automaton from $|M_2|$ to $|P'_2|$ is at most $c(k^3 + 2k^2 + k)$. So that player 2 does not profit by this deviation, we need to require that

$$x_2^* \geq 1 + \frac{4}{k_2} + \frac{8}{k} + c(k^3 + 2k^2 + k),$$

and therefore it is enough to require that

$$x_2^* - 1 > \eta > \frac{4}{k_2} + \frac{8}{k} + c(k^3 + 2k^2 + k).$$

The right-hand side inequality holds provided

$$c < \frac{\eta - \frac{4}{k_2} - \frac{8}{k}}{k^3 + 2k^2 + k},$$

so it is enough to require that $c < \frac{\eta}{2k^3}$. ■

We finally prove that player 2 cannot profit by deviating to an automaton larger than M_2 .

Lemma 11 *Let P'_2 be a pure automaton such that $\gamma_2(M_1(k), P'_2) > x_2^*$. Then $\gamma_2(M_1, P'_2) - c|P'_2| \leq \gamma_2(M_1, M_2) - c|M_2|$, provided $c > \frac{3}{k^3 k_2}$.*

Proof. Let L_0 be the number of pure automata (P_1^l) that P_2 fools. Because $\gamma_2(M_1, P'_2) > x_2^*$ we have $L_0 \geq 1$. If P_2 fools P_1^l , player 2's long-run average payoff is at most 4, the maximal payoff in the game. If P_2 does not fool P_1^l , player 1's long-run average payoff is at most x_2^* . The expected long-run average payoff of player 2 then satisfies

$$\gamma_2(M_1, P'_2) \leq 4 \frac{L_0}{k_2} + x_2^* \frac{k_2 - L_0}{k_2} < x_2^* + 3 \frac{L_0}{k_2}.$$

By Corollary 9 we have $|P'_2| \geq L_0 k^3$, and therefore

$$\gamma_2(M_1, P'_2) < x_2^* + 3 \frac{L_0}{k_2} = x_2^* + 3 \frac{L_0 k^3}{k^3 k_2} \leq x_2^* + |P'_2| \times \frac{3}{k^3 k_2}.$$

Therefore, as soon as $c > \frac{3}{k^3 k_2}$ player 2 does not profit by this deviation. ■

To summarize, given a feasible and an individually rational payoff vector x^* , we choose $\eta \in (0, \min\{x_1^* - 1, x_2^* - 1\})$. Let $c > 0$ be sufficiently small, and let $k = k_c$ satisfy $\frac{3}{k^3 k_2} < c < \frac{\eta}{3k^3}$. Then the automata $(M_1(k), M_2(k))$ form a c -BCC equilibrium. Since the size of the automata $M_1(k)$ and $M_2(k)$ are $k^3 + 2k^2 + 1$ and $k^3 + 2k^2 + k + 1$, if for each $k \geq 1$ we set $\hat{c}_k = \frac{4}{k^3 k_2}$, then $\frac{3}{k^3 k_2} < \hat{c}_k < \frac{\eta}{2k^3}$ and $\hat{c}_k M_1(k)$ and $\hat{c}_k M_2(k)$ are both smaller than $\frac{10}{k_2}$, which goes to 0 as k goes to infinity (and \hat{c}_k goes to 0). It follows that x^* is a BCC equilibrium payoff.

4 The General Case

In Section 3 we proved Theorem 4 for the Prisoner's Dilemma. In the present section we explain how the proof should be adapted to prove the result for arbitrary games. In the play path ω^* , the punishment phase, as well as the regular play, are similar to those in Section 3, and only the babbling phase significantly changes.

Assume w.l.o.g. that payoffs are bounded by 1, and let $x \in F \cap V$. To rule out trivial cases, assume that each player has at least two actions. The vector x is a convex combination of *all* the entries in the payoff matrix

$$\left| x - \sum_{a \in A} \alpha_a u(a) \right| \leq \varepsilon,$$

where $(\alpha_a)_{a \in A}$ are non-negative numbers summing to 1. In fact, by Caratheodory's Theorem, x is a convex combination of three entries in the payoff matrix. Instead of handling separately each of the alternative configurations of these three entries, we find it simpler to handle the general case.

Fix $\varepsilon > 0$, a natural number $k_0 > \frac{1}{\varepsilon}$, and a natural number k . Let $(k_a)_{a \in A}$ be a collection of positive integers such that (a) $\sum_{a \in A} k_a = k_0$, and (b) $|k_a - \alpha_a k_0| \leq 1$. Define the regular path

$$\omega_0 = \sum_{a=(a_1, a_2) \in A} k_a \times (a_1, a_2).$$

Then the average payoff along ω_0 is within ε of x .

For each $i = 1, 2$, denote by $l_i = |A_i|$ the number of actions of player i , and by $A_i = \{a_i^1, a_i^2, \dots, a_i^{l_i}\}$ her actions. Assume w.l.o.g. that $l_1 \leq l_2$, and that a_i^1 is the min-max strategy of player i against player $3 - i$.

The play path ω^* is defined as follows:

$$\begin{aligned} \omega^* = & k^4 \times (a_1^1, a_2^1) + \left(\sum_{j=1}^{k^2} \sum_{m=1}^{l_1} k \times (a_1^m, a_2^m) \right) + (k+1) \times (a_1^1, a_2^1) \\ & + \left(\sum_{j=1}^k \sum_{m=l_1+1}^{l_2} k \times (a_1^1, a_2^m) \right) + (a_1^2, a_2^1) + \sum_{j=1}^{\infty} \omega_0. \end{aligned}$$

Both the punishment phase and the babbling phase are longer in this construction than in the construction for the Prisoner's Dilemma, yet the punishment phase is much longer, to ensure that the payoff that results from a deviation is close to the min-max value in pure strategies.

The complexity of ω^* w.r.t. player 1 is $k^4 + k^3 l_1 + 1$. Indeed, as in Section 3, the complexity of the prefix $\omega^*(1) = k^4 \times (a_1^1, a_2^1) + \left(\sum_{j=1}^{k^2} \sum_{m=1}^{l_1} k \times (a_1^m, a_2^m) \right) + (k+1) \times (a_1^1, a_2^1)$ is $k^4 + k^3 l_1 + 1$, and to implement the rest of the play ω^* player 1 can re-use states that were used to implement $\omega^*(1)$. One can verify that the complexity of ω^* w.r.t. player 2 is $k^4 + k^3 l_1 + (k+1) + k^2(l_2 - l_1)$. A similar construction of automata M_1 and M_2 for the two players, that re-uses states to implement the rest of ω^* , shows that x is a BCC equilibrium payoff.

References

- [1] D. Abreu and A. Rubinstein. (1988) The structure of Nash equilibrium in repeated games with finite automata, *Econometrica* 56, 1259-1281.
- [2] J. Banks, and R. Sundaram (1990) Repeated games, finite automata and complexity, *Games and Economic Behaviour*, 2, 97-117.
- [3] E. Ben Porath, (1993), Repeated games with finite automata, *Journal of Economic Theory*, 59, 17-32.
- [4] K. Chatterjee, and H. Sabourian (2000), Multiperson bargaining and strategic complexity, *Econometrica*, 68, 1491-1509.
- [5] K. Chatterjee, and H. Sabourian (2008), Game Theory and Strategic Complexity, in *Encyclopedia of Complexity and System Science*, Editor-in-Chief Robert A. Meyers, Springer.
- [6] D. Gale, and H. Sabourian, (2005) Complexity and competition, *Econometrica*, 73, 739-770.
- [7] Halpern J.Y. and Pass R. (2008) Game Theory with Costly Computation, preprint.
- [8] E. Kalai (1990) Bounded Rationality and Strategic Complexity in Repeated Games, in *Game Theory and Applications*, eds. Ichiishi, Neyman and Tauman, San Diego: Academic Press, 1990, 131-157.
- [9] E. Maenner (2008) Adaptation and complexity in repeated games, *Games and Economic Behavior*, 63, 166-187.
- [10] A. Neyman, (1985) Bounded complexity justifies cooperation in the finitely-repeated Prisoners' Dilemma, *Economics Letters*, 19, 227-229.
- [11] A. Neyman, (1997) Cooperation, repetition and automata, in *Cooperation: Game-Theoretic Approaches*, NATO ASI Series F, Vol. 155, S. Hart and A. Mas-Colell (eds.), Springer-Verlag. 233-255.
- [12] A. Neyman (1998) Finitely repeated games with finite automata, *Mathematics of Operations Research*, 23, 513-552
- [13] A. Neyman, and D. Okada (1999) Strategic entropy and complexity in repeated games. *Games and Economic Behavior*, 29, 191-223.

- [14] A. Neyman, and D. Okada (2000) Repeated games with bounded entropy. *Games and Economic Behavior*, 30, 228-247.
- [15] A. Neyman, and D. Okada, (2000) Two-person repeated games with finite automata. *International Journal of Game Theory*, 29, 309-325.
- [16] M. Piccione (1992) Finite automata equilibria with discounting, *Journal of Economic Theory*, 56, 180-193.
- [17] M. Piccione, and A. Rubinstein (1993) Finite automata play a repeated extensive game, *Journal of Economic Theory*, 61, 160-168.
- [18] A. Rubinstein (1986) Finite automata play the repeated prisoner's dilemma, *Journal of Economic Theory*, 39, 83-96.
- [19] A. Rubinstein, (1998) *Modeling bounded rationality*, MIT Press, Cambridge, Mass.
- [20] H. Sabourian, (2003) Bargaining and markets: complexity and the competitive outcome, *Journal of Economic Theory*, 116 , 189-228.
- [21] H.A.Simon, (1972) Theories of bounded rationality, in "Decision and Organization" (C.B. McGuire and R. Radner, Eds), North- Holland, Amsterdam.
- [22] H.A. Simon,(1978) On how to decide what to do, *Bell Journal of Economics*, 9, 494-507.
- [23] E. Zemel (1989) Small talk and cooperation: A note on bounded rationality, *Journal of Economic Theory*, 49, 1-9.